APPARATUS AND METHOD FOR SELECTING PROGRAM
HALTS IN AN UNPROTECTED PIPELINE AT NON-
INTERRUPTIBLE POINTS IN CODE EXECUTION

This application claims priority under 35 USC §119(e)(1) of
Provisional Application Number 60/434,231 (TI-34657P) filed
December 17, 2002.

Related Applications

5   U.S. Patent Application (Attorney Docket No. TI-34654),
entitled APPARATUS AND METHOD FOR SYNCHRONIZATION OF TRACE
STREAMS FROM MULTIPLE PROCESSORS, invented by Gary L.
Swoboda, filed on even date herewith, and assigned to the
assignee of the present application; U.S. Patent
10 Application (Attorney Docket No. TI-34655), entitled
APPARATUS AND METHOD FOR SEPARATING DETECTION AND ASSERTION
OF A TRIGGER EVENT, invented by Gary L. Swoboda, filed on

5 even date herewith, and assigned to the assignee of the
   present application; U.S. Patent Application (Attorney
   Docket No. TI- 34656), entitled APPARATUS AND METHOD FOR
   STATE SELECTABLE TRACE STREAM GENERATION, invented by Gary
   L. Swoboda, filed on even date herewith, and assigned to
10 the assignee of the present application; U.S. Patent
   Application (Attorney Docket No. TI-34658), entitled
   APPARATUS AND METHOD FOR REPORTING PROGRAM HALTS IN AN
   UNPROTECTED PIPELINE AT NON-INTERRUPTIBLE POINTS IN CODE
   EXECUTION, invented by Gary L. Swoboda, filed on even date
15 herewith, and assigned to the assignee of the present
   application; U.S. Patent Application (Attorney Docket No.
   TI-34659), entitled APPARATUS AND METHOD FOR A FLUSH
   PROCEDURE IN AN INTERRUPTED TRACE STREAM, invented by Gary
   L. Swoboda, filed on even date herewith, and assigned to
20 the assignee of the present application; U.S. Patent
   Application (Attorney Docket No. TI-34660), entitled
   APPARATUS AND METHOD FOR CAPTURING AN EVENT OR COMBINATION
   OF EVENTS RESULTING IN A TRIGGER SIGNAL IN A TARGET
   PROCESSOR, invented by Gary L. Swoboda, filed on even date
25 herewith, and assigned to the assignee of the present
   application; U.S. Patent Application (Attorney Docket No.
   TI-34661), entitled APPARATUS AND METHOD FOR CAPTURING THE
   PROGRAM COUNTER ADDRESS ASSOCIATED WITH A TRIGGER SIGNAL IN
   A TARGET PROCESSOR, invented by Gary L. Swoboda, filed on
30 even date herewith, and assigned to the assignee of the
   present application; U.S. Patent Application (Attorney

5   Docket No. TI-34662), entitled APPARATUS AND METHOD
     DETECTING ADDRESS CHARACTERISTICS FOR USE WITH A TRIGGER
     GENERATION UNIT IN A TARGET PROCESSOR, invented by Gary
     Swoboda and Jason L. Peck, filed on even date herewith, and
     assigned to the assignee of the present application; U.S.
10   Patent Application (Attorney Docket No. TI-34663), entitled
     APPARATUS AND METHOD FOR TRACE STREAM IDENTIFICATION OF A
     PROCESSOR RESET, invented by Gary L. Swoboda, Bryan Thome
     and Manisha Agarwala, filed on even date herewith, and
     assigned to the assignee of the present application; U.S.
15   Patent (Attorney Docket No. TI-34664), entitled APPARATUS
     AND METHOD FOR TRACE STREAM IDENTIFICATION OF A PROCESSOR
     DEBUG HALT SIGNAL, invented by Gary L. Swoboda, Bryan
     Thome, Lewis Nardini and Manisha Agarwala, filed on even
     date herewith, and assigned to the assignee of the present
20   application; U.S. Patent Application (Attorney Docket No.
     TI-34665), entitled APPARATUS AND METHOD FOR TRACE STREAM
     IDENTIFICATION OF A PIPELINE FLATTENER PRIMARY CODE FLUSH
     FOLLOWING INITIATION OF AN INTERRUPT SERVICE ROUTINE;
     invented by Gary L. Swoboda, Bryan Thome and Manisha
25   Agarwala, filed on even date herewith, and assigned to the
     assignee of the present application; U.S. Patent
     Application (Attorney Docket No. TI-34666), entitled
     APPARATUS AND METHOD FOR TRACE STREAM IDENTIFICATION OF A
     PIPELINE FLATTENER SECONDARY CODE FLUSH FOLLOWING A RETURN
30   TO PRIMARY CODE EXECUTION, invented by Gary L. Swoboda,
     Bryan Thome and Manisha Agarwala filed on even date

5 herewith, and assigned to the assignee of the present
   application; U.S. Patent Application (Docket No. TI-34667),
   entitled APPARATUS AND METHOD IDENTIFICATION OF A PRIMARY
   CODE START SYNC POINT FOLLOWING A RETURN TO PRIMARY CODE
   EXECUTION, invented by Gary L. Swoboda, Bryan Thome and
10 Manisha Agarwala, filed on even date herewith, and assigned
   to the assignee of the present application; U. S. Patent
   Application (Attorney Docket No. TI-34668), entitled
   APPARATUS AND METHOD FOR IDENTIFICATION OF A NEW SECONDARY
   CODE START POINT FOLLOWING A RETURN FROM A SECONDARY CODE
15 EXECUTION, invented by Gary L. Swoboda, Bryan Thome and
   Manisha Agarwala, filed on even date herewith, and assigned
   to the assignee of the present application; U.S. Patent
   Application (Attorney Docket No. TI-34669), entitled
   APPARATUS AND METHOD FOR TRACE STREAM IDENTIFICATION OF A
20 PAUSE POINT IN A CODE EXECTION SEQUENCE, invented by Gary
   L. Swoboda, Bryan Thome and Manisha Agarwala, filed on even
   date herewith, and assigned to the assignee of the present
   application; U.S. Patent Application (Attorney Docket No.
   TI-34670), entitled APPARATUS AND METHOD FOR COMPRESSION OF
25 A TIMING TRACE STREAM, invented by Gary L. Swoboda and
   Bryan Thome, filed on even date herewith, and assigned to
   the assignee of the present application; U.S. Patent
   Application (Attorney Docket No. TI-34671), entitled
   APPARATUS AND METHOD FOR TRACE STREAM IDENTIFCATION OF
30 MULTIPLE TARGET PROCESSOR EVENTS, invented by Gary L.
   Swoboda and Bryan Thome, filed on even date herewith, and

5   assigned to the assignee of the present application; and

       U.S. Patent Application (Attorney Docket No. TI-34672

       entitled APPARATUS AND METHOD FOR OP CODE EXTENSION IN

       PACKET GROUPS TRANSMITTED IN TRACE STREAMS, invented by

       Gary L. Swoboda and Bryan Thome, filed on even date

10   herewith, and assigned to the assignee of the present

       application are related applications.

Background of the Invention

15

1.   Field of the Invention

       This invention relates generally to the testing of digital

       signal processing units and, more particularly, to the

20   interruption of code execution to determine the status of

       various portion of the target processor implementing the

       code or to initiate a different code execution routine. A

       processor can have a protected pipeline or a non-protected

       pipeline.   When the target processor has a non-protected

25   pipeline, the code executing on the processor can have

       interruptible portions and can have non-interruptible

       portions.

2.   Description of the Related Art

30

       As microprocessors and digital signal processors have

       become increasingly complex, advanced techniques have been

5   developed to test these devices.   Dedicated apparatus is
    available to implement the advanced techniques.   Referring
    to Fig. 1A, a general configuration for the test and debug
    of a target processor 12 is shown.   The test and debug
    procedures operate under control of a host processing unit

10 **10**.   The host processing unit **10** applies control signals to
    the emulation unit **11** and receives (test) data signals from
    the emulation unit **11** by cable connector **14**.   The emulation
    unit **11** applies control signals to and receives (test)
    signals from the target processor **12** by connector cable **15**.

15 The emulation unit **11** can be thought of as an interface
    unit between the host processing unit **10** and the target
    processor **12**.   The emulation unit **11** must process the
    control signals from the host processor unit **10** and apply
    these signals to the target processor **12** in such a manner

20 that the target processor will respond with the appropriate
    test signals.   The test signals from the target processor
    12 can be a variety types.   Two of the most popular test
    signal types are the JTAG (Joint Test Action Group) signals
    and trace signals.   The JTAG signal provides a standardized

25 test procedure in wide use.   Trace signals are signals from
    a multiplicity of junctions in the target processor **12**.
    While the width of the bus interfacing to the host
    processing unit **10** generally have a standardized width, the
    bus between the emulation unit **11** and the target processor

30 **12** can be increased to accommodate the increasing
    complexity of the target processing unit **12**.   Thus, part of

5   the interface function between the host processing unit **10**
    and the target processor **12** is to store the test signals
    until the signals can be transmitted to the host processing
    unit **10**.

10 In the test and debug of the target processor, specified
    internal events result in a halt of the target processor
    (i.e., for analysis of the configuration of the processor)
    or in a change of processor program execution. These
    specified events are monitored by dedicated apparatus.
15 Upon detection of the occurrence of the event, the
    monitoring apparatus generates an event signal. The events
    signal or signals are applied to a trigger device. The
    trigger device issues a trigger signal that results in the
    change of operation of the target processor. Referring to
20 Fig. 1B, the operation of the trigger generation unit **19** is
    shown.   Monitoring apparatus **18**, including event signal
    generation units **181** through **18N**, is typically included in
    the target processor **12**. The event generation units **181** –
    18N each monitors some portion of the target processor to
25 determine when a specified condition or event is present.
    When the specified condition is detected by the event
    signal generation unit monitoring the condition, an event
    signal is generated. The event signals are applied to the
    trigger generation unit **19**.   Based on the event signals
30 applied to the trigger generation unit **19**, a trigger signal
    is selected.   Certain events and combination of events,

5   referred to as an event front, generate a selected trigger
     signal that results in certain activity in the target
     processor, e.g. a debug halt. Combinations of different
     events generating trigger signals are referred to as jobs.
     Multiple jobs can result in the same trigger signal or
10   combination of trigger signals. In the test and debug of
     the target processor, the trigger signals can provide
     impetus for changing state in the target processor or for
     performing a specified activity. The event front defines
     the reason for the generation of trigger signal. This
15   information is important in understanding the operation of
     the target processor because, as pointed out above, several
     combinations of events can result in the generation of a
     trigger signal. In order to analyze the operation of the
     target processing unit, the portion of the event front
20   resulting in the trigger signal must be identified in order
     to determine the reason for the generation of the trigger
     signal.

     A development system can create a number of test and debug
25   events. These test and debug events halt the code
     execution so that analysis can be made of the state of the
     processor. In a real-time test and debug environment, it
     is desirable to allow the service of interrupt signals
     designated as real time interrupts to continue after a
30   debug event generates an execution halt. Because the test
     and debug events are generally accepted at the next

5   instruction boundary, a test and debug event can halt the
code execution at a non-interruptible point in the code
execution.

In a protected pipeline, real-time interrupt procedures can
10   occur at any instruction boundary, so it is not a problem
that code execution halts in a non-interruptible point.
Once the code execution is halted, real-time interrupt
procedures continue even though the code was not
interruptible at the point at which the code execution was
15   halted.   In other words, in a non-interruptible code
portion, real time interrupt procedures can continue in a
protected pipeline independent of whether code execution is
halted at a non-interruptible point.

20   In an unprotected pipeline, the situation is much different
than for a protected pipeline.   In the unprotected
pipeline, real time interrupts cannot occur at any
arbitrary instruction boundary because of architectural
problems (e.g., delayed branches in flight) or instruction-
25   to-instruction relationships that can be disturbed (the
global enable bit is disabled to indicate these code
areas).   Because the pipeline sequence must be preserved in
an unprotected pipeline, this rule must also be obeyed when
code execution is halted by a test and debug event.

30

5   When a test and debug event is allowed to halt code
     execution in an unprotected pipeline at a non-interruptible
     point in the code execution, real time interrupt services
     must be blocked because these activities would corrupt the
     code so that the code execution could not be resumed after
10   the interrupt return. To preserve the ability to service
     real-time interrupts after code execution halts, test and
     debug events must be blocked until code execution reaches
     an interruptible point.

15   However, an application developer may find it desirable to
     halt the code execution at a non-interruptible point in the
     code to observe the machine state even though real-time
     interrupt are blocked, and other times, may find it
     desirable to delay code execution halts to points where the
20   code is interruptible (i.e., allowing service of real time
     interrupts after execution halts.

     A need has therefore been felt for apparatus and an
     associated method having the feature that a program
25   execution halt can be taken in a non-protected pipeline
     during a non-interruptible portion of the code. It would
     be further feature of the apparatus and associated method
     to permit a user to select whether a program execution halt
     can be performed in a protected pipeline during execution
30   of non-interruptible code portion. It would be yet another
     feature of the apparatus and the associated method to

5   permit a program execution halt during either a protected
    portion of the program execution or an unprotected portion
    of the program code execution in a protected pipeline.

5   **Summary of the Invention**

The aforementioned and other features are accomplished, according to the present invention, by providing a storage unit for storing a signal indicating that the program 10   execution halt can be implemented in an unprotected pipeline whether the code is interruptible or non-interruptible. When the signal is present in the storage unit, a halt request will be forwarded immediately thereby resulting in a code execution halt. When the signal is not 15   present in the storage unit, the halt request signal will be forwarded only during an interruptible portion of the code execution. The signal can be stored in the storage unit by the program or by the intervention through the test and debug facilities.
20

Other features and advantages of present invention will be more clearly understood upon reading of the following description and the accompanying drawings and the claims.

25   **Brief Description of the Drawings**

Figure 1A is a block diagram of the apparatus used in the test and debug of a target processor; while Figure 1B illustrates the generation of trigger signals.
30

5   Figure 2 is a block diagram of the apparatus for forwarding a halt signal during a non-interruptible code execution portion in a non-protected pipeline according to the present invention.

10 **Description of the Preferred Embodiment**

1. Detailed Description of the Figures

Fig. 1A and Fig. 1B have been discussed with respect to the
15 related art.

Referring to Fig. 2, the apparatus for selectively halting code execution in a processor having a non-protected pipeline is shown. Storage unit **20** has a CONTROL signal
20 applied thereto. When the CONTROL signal is stored in the storage unit **20**, a signal is applied to a first terminal of logic OR gate **22**. A second terminal of logic OR gate **22** has a signal indicating whether the executing code is currently in an interruptible or in a non-interruptible
25 code portion. The output terminal of logic OR gate **22** is coupled to a first input terminal of logic AND gate **21**. A second terminal of logic AND gate **21** has a HALT REQUEST signal applied thereto. A HALT signal is generated at the output terminal of logic AND gate **21**.
30

2. Operation of the Preferred Embodiment

5   The operation of the present invention can be understood as
     follows.   In a processing system having a non-protected
     pipeline, when a CONTROL signal is not stored in the
     storage unit **20** and a HALT REQUEST signal is applied to the
     second terminal of the logic AND gate **21,** then a HALT
10   signal will be applied to the output terminal of logic AND
     gate **21** only when a positive INTERRUPTIBLE CODE PORTION
     signal is applied to the second input terminal of logic OR
     gate **22.**   When the INTERRUPTIBLE CODE PORTION signal and
     the CONTROL signal are not present, then the HALT REQUEST
15   signal will not result in a HALT signal.   However, when the
     CONTROL signal is stored in storage unit **20,** a CONTROL
     signal is applied to an input terminal of logic OR gate **22**
     and a signal is applied to the first input terminal of
     logic AND gate **21.**   In this situation, a HALT REQUEST
20   signal will provide a HALT signal whether the INTERUPTIBLE
     CODE PORTION signal is present or not.

     In this manner, a HALT signal can be generated even when a
     non-interruptible   code   portion   is   being   executed.
25   Furthermore, the code execution in a non-interruptible code
     portion is determined by the storage of the CONTROL signal
     in the storage unit.   Therefore, the generation of a HALT
     REQUEST signal is under the control of the user testing or
     debugging the target processor.

30

5   While the invention has been described with respect to the embodiments set forth above, the invention is not necessarily limited to these embodiments. Accordingly, other embodiments, variations, and improvements not described herein are not necessarily excluded from the

10 scope of the invention, the scope of the invention being defined by the following claims.